CMPSCI 250: Introduction to Computation

Lecture #3: Set Operations and Truth Table Proofs

David Mix Barrington

27 January 2012

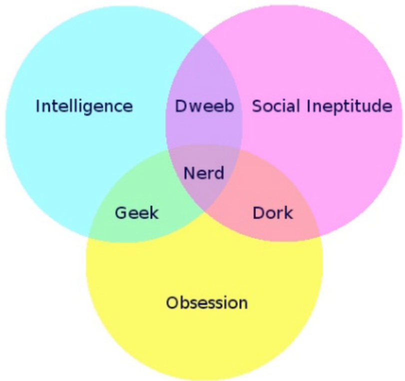
Set Operations and Truth Table Proofs

- Venn Diagrams
- Carroll Diagrams
- Set Operations
- Propositions About Sets
- The Setting For Propositional Proofs
- How to Do a Truth Table Proof
- A Truth Table Proof Example

Sets and Venn Diagrams

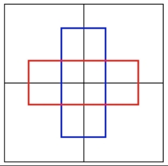
- Suppose we have multiple sets whose elements all come from a single type.
- Each set divides the type into two groups -- the elements in the set and the elements not in the set.
- Two sets give us four total groups, three sets give us eight, four sets give 16, and so forth -- k sets make 2^k total groups.
- A **Venn diagram** can represent these groups, as with the three sets at left. On the homework, you'll draw a general Venn diagram for four sets.

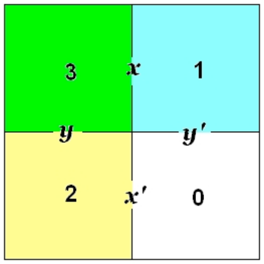
A Venn Diagram From cubiclebot.com

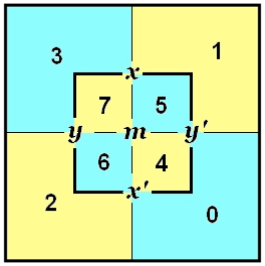


Carroll Diagrams

- Lewis Carroll (author of *Alice in Wonderland*) was a contemporary of Venn and had his own system of diagrams.
- The top diagram represents the four possible combinations of being in the set x or y. For example, region 2 is in y but not in x.
- The bottom diagram includes a third set m, inside the central box. Region 5 is in m and x but not in y. Note the binary for 5, 101, codes these three bits: yes-m, no-y, yes-x.
- What about four sets?







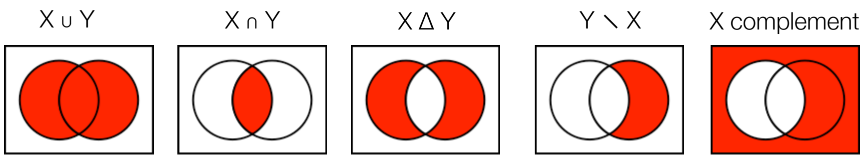
Diagrams from hom.wikidot.com

Set Operations

- We have a number of **binary operations** on sets, that each take two sets as input and give one set as output.

- If X and Y are sets, their **union** X ∪ Y is the set of all elements in either X or Y, and their **intersection** X ∩ Y is the set of all elements that are in both.

- The **symmetric difference** X Δ Y is the set of elements in *exactly one of* X and Y. The **relative complement** X ∖ Y is the elements in X, but not in Y. The **complement** of X (X with a line over it) is the set of elements not in X.



Diagrams from wikipedia.org, “Venn Diagram”

Propositions About Sets

- Given two sets X and Y, we can form the propositions $X = Y$ and $X \subseteq Y$. We can also use the = and $\subseteq$ operators on more complicated sets formed with the set operators, for example $(X \setminus Y) \cap (Y \setminus X) = \emptyset$.

- This last statement is an example of a **set identity** because it is true no matter what the sets X and Y are. Since all the elements of $X \setminus Y$ are in X, and none of the elements of $Y \setminus X$ are in X, no element could be in both.

- Equality and subset statements about sets are actually compound propositions involving **membership statements** for the original sets. For example, $X = Y$ means that for any object z of the correct type, the propositions $z \in X$ and $z \in Y$ are either both true or both false: $z \in X \leftrightarrow z \in Y$.

- Similarly, $X \subseteq Y$ means that for any z, $z \in X$ implies $z \in Y$: $z \in X \rightarrow z \in Y$.

Set Identities With Set Operators

- A set statement like $(X \setminus Y) \cap (Y \setminus X) = \varnothing$, using set operations and the equality or subset operator, can be translated into a compound proposition.

- We want to say $z = (X \setminus Y) \cap (Y \setminus X) \leftrightarrow z \in \varnothing$. But the statement on the left of the $\leftrightarrow$ can be simplified, to $z \in (X \setminus Y) \wedge z \in (Y \setminus X)$. And using the definition of $\setminus$, this can be further simplified to $(z \in X \wedge \neg (z \in Y)) \wedge (z \in Y \wedge \neg(z \in X))$.

- If we define the boolean x to mean $z \in X$ and the boolean y to mean $z \in Y$, we can rewrite the whole statement as $(x \wedge \neg y) \wedge (y \wedge \neg x) \leftrightarrow 0$, where we use 0 to mean the false proposition. This compound proposition is a tautology.

- In the same way we can translate any set statement, because each set operation corresponds exactly to a boolean operation on membership statements.

The Setting for Propositional Proofs

- The propositional calculus lets us form compound propositions from atomic propositions, and then ask questions about them.

- Is a given statement P a **tautology**? If we know that a **premise** statement P is true, does that guarantee that another **conclusion** statement C is also true? Given two statements P and Q, are they **equivalent**?

- Verifying tautologies solves all three of these questions, because they ask whether P, $P \rightarrow C$, and $P \leftrightarrow Q$ respectively are tautologies.

- In this lecture we'll see how to verify a tautology with a **truth table**.

- Next week we'll see how to verify that an implication or an equivalence is a tautology with a **deductive sequence proof** or an **equational sequence proof**.

How to Do a Truth Table Proof

- The idea of a truth table proof is that if we have k atomic propositions, there are 2^k possible settings of the truth values of those propositions. If a given compound proposition is true in all of those cases, it is a tautology.

- We need to evaluate the compound proposition systematically, in all the cases. We begin by listing the cases, which we can do by **counting in binary** from 0 to 2^k - 1, which is from 00...0 to 11...1. (This is much less error-prone than trying to get all the cases in some arbitrary order.)

- The basic idea is that *under* each symbol of the compound proposition, we will have a column of 2^k 0's and 1's to represent the values, in each case, of the compound proposition associated with that symbol.

- We begin with the occurrences of the variables, then calculate new columns in the order that operations are used to evaluate the compound proposition.

A Truth Table Example

- Let's take the formula (x ∧ ¬ y) ∧ (y ∧ ¬ x) ↔ 0. There are four cases 00, 01, 10, and 11, where the first bit is the truth value of x and the second that of y. We write the correct column under each occurrence of a variable. We also write a column of all 0's under the 0, since this symbol always has the value 0.

```
x y | (x ∧ ¬ y) ∧ (y ∧ ¬ x) ↔  0
--------------------------------
0 0    0     0     0     0     0
0 1    0     1     1     0     0
1 0    1     0     0     1     0
1 1    1     1     1     1     0
```

Continuing the Example

- Next we fill in the columns for the ¬ operations:

x	y	(x	∧	¬	y)	∧	(y	∧	¬	x)	↔	0
0	0	0		**1**	0		0		**1**	0		0
0	1	0		**0**	1		1		**1**	0		0
1	0	1		**1**	0		0		**0**	1		0
1	1	1		**0**	1		1		**0**	1		0

- Then the two ∧ operations inside the parentheses:

x	y	(x	∧	¬	y)	∧	(y	∧	¬	x)	↔	0
0	0	0	**0**	1	0		0	**0**	1	0		0
0	1	0	**0**	0	1		1	**1**	1	0		0
1	0	1	**1**	1	0		0	**0**	0	1		0
1	1	1	**0**	0	1		1	**0**	0	1		0

Finishing the Example

- Then the last ∧ operation:

```
x y | (x ∧ ¬ y) ∧ (y ∧ ¬ x) ↔  0
--------------------------------
0 0    0 0 1 0  0  0 0 1 0     0
0 1    0 0 0 1  0  1 1 1 0     0
1 0    1 1 1 0  0  0 0 0 1     0
1 1    1 0 0 1  0  1 0 0 1     0
```

- And finally the ↔ operation. Since this final column is all 1's, we have shown that the original compound proposition is a tautology.

```
x y | (x ∧ ¬ y) ∧ (y ∧ ¬ x) ↔  0
--------------------------------
0 0    0 0 1 0  0  0 0 1 0  1  0
0 1    0 0 0 1  0  1 1 1 0  1  0
1 0    1 1 1 0  0  0 0 0 1  1  0
1 1    1 0 0 1  0  1 0 0 1  1  0
```